Algoritmo Quicksort: cómo implementar en Python

B

brendomatos

13/06/2022

> *Ya [hemos implementado la solución de clasificación de listas de estudiantes] de dos maneras diferentes y hemos visto que MergeSort funciona mucho mejor en comparación con SelectionSort.*

Entonces, ¿es MergeSort la mejor solución de clasificación que existe?

Ahora implementemos otra solución y luego haremos una pequeña comparación.

Seguiremos los siguientes pasos para alcanzar nuestro objetivo:

Elegiremos un pivote (en este caso, el primer elemento de la lista) y cambiaremos su posición con el elemento en el medio; Repasemos toda la lista y verifiquemos elemento por elemento, comparándolos con el pivote. A partir de entonces: Si el ítem está en una posición más baja que el pivote en orden alfabético, será transferido o mantenido en la lista

de la izquierda; Si el elemento está en una posición superior al pivote en orden alfabético, se transferirá o se mantendrá en la lista de la derecha. Haciendo esto recursivamente, al final tendremos una lista ordenada

¿Lo Implementamos?

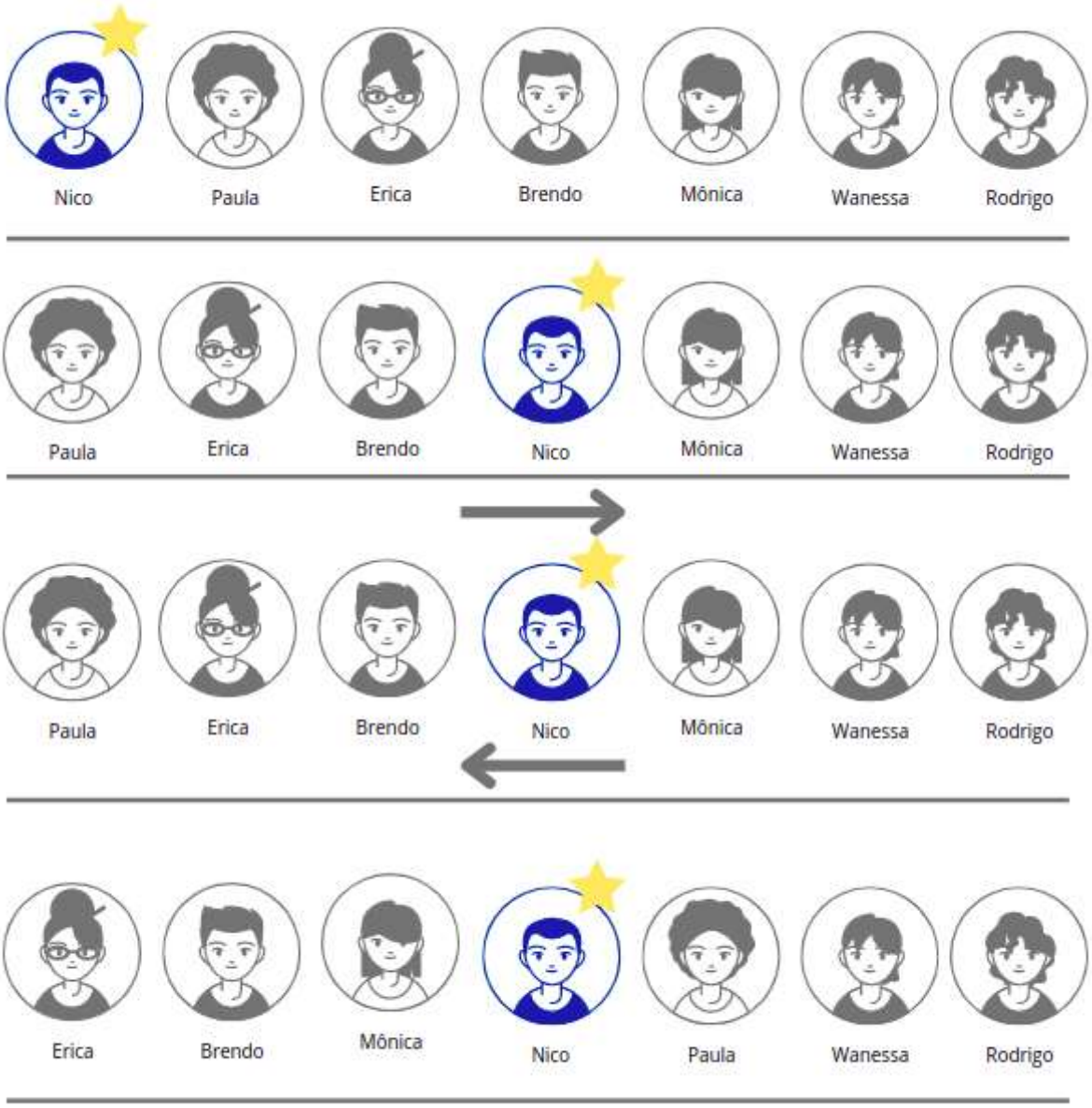
```
def importar_lista(archivo):
    lista = []
    with open(archivo) as tf:
        lines = tf.read().split('","')
    for line in lines:
        lista.append(line)
    return lista

def ordenar(lista):
    tamano_de_lista = len(lista)
    if tamano_de_lista > 0:
        quick_sort(lista, 0, tamano_de_lista - 1)

def quick_sort(lista, inicio, fin):
    if inicio > fin:
        return
    anterior = inicio
    posterior = fin
    pivo = lista[inicio]

    while anterior < posterior:
        while anterior < posterior and lista[posterior] > pivo:
            posterior = posterior - 1

        if anterior < posterior:
            lista[anterior] = lista[posterior]
            anterior = anterior + 1

        while anterior < posterior and lista[anterior] <= pivo:
            anterior = anterior + 1
```

```
            if anterior < posterior:
                lista[posterior] = lista[anterior]
                posterior = posterior - 1

            lista[anterior] = pivo

        quick_sort(lista, inicio, anterior - 1)
        quick_sort(lista, anterior + 1, fin)

    def main():
        lista_de_alumnos = importar_lista('../data/lista_alumnos')

        ordenar(lista_de_alumnos)

        for nombre in lista_de_alumnos:
            print(nombre)

    if __name__ == "__main__":
        main()
```

Notamos que el rendimiento con el que se desempeñó QuickSort es muy similar al de MergeSort, ¿no es así? ¿Y la complejidad? ¿Los comparamos?

Al igual que en MergeSort, dividimos la lista recursivamente, pero no necesariamente a la mitad, sino desde un pivote. Esto nos recuerda la notación O(lg N), ¿verdad?

Además, debemos tener en cuenta que, con cada llamada recursiva, repasamos toda la lista para asegurarnos de que todos los alumnos cuyos nombres estén en una posición inferior al pivote en orden alfabético estén a su izquierda. Los estudiantes con nombres más altos que el pivote en orden alfabético deben estar a su derecha (como se muestra en la imagen a continuación), lo que nos recuerda a un algoritmo lineal, ¿verdad? Una vez más, absolutamente. Entonces podemos determinar que esto nos da la complejidad O(N lg N) al igual que en MergeSort.



Ahora, ¿el algoritmo QuickSort se ejecuta en O(N lg N) en cualquier escenario? ¿Cuáles serían las diferencias de eficiencia entre QuickSort y MergeSort, ya que siempre parecen tener la misma complejidad? ¿Cuál es el mejor algoritmo?

Explicamos con más profundidad esta comparación entre los algoritmos QuickSort y MergeSort después de comprender mejor la Notación BigO.

Brendo Rodrigo Souza de Matos

Ingeniero de Software apasionado por lo que hace, amante de los nuevos retos y sediento de conocimiento. Actualmente soy Ingeniero de Software de Plataformas en Méliuz (B3: CASH3) y estoy realizando una Maestría en Ciencias de la Computación en la Universidad Federal de Amazonas.

Este articulo fue adecuado para Alura Latam por: Wilfredo Rojas

Cursos de Programación

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

En Alura encontrarás variados cursos sobre Programación. ¡Comienza ahora!

SEMESTRAL

US$49,90

un solo pago de US$49,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

Acceso a todos
los cursos

Estudia las 24 horas,
dónde y cuándo quieras

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES
INSTRUCTORES
BLOG
POLÍTICA DE PRIVACIDAD
TÉRMINOS DE USO
SOBRE NOSOTROS
PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN
FRONT END
DATA SCIENCE
INNOVACIÓN Y GESTIÓN
DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES

ALIADOS

Empresa participante do SCALE UP ENDEAVOR

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.

Google for Startups Alumni 2021

Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science
Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps
Docker | Linux

Cursos de Innovación y Gestión
Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento